# Large-Scale Stochastic Mixed-Integer Programming Algorithms for Power Generation Scheduling

Kibaek Kim and Victor M. Zavala

**Abstract** This chapter presents a stochastic unit commitment model for power systems and revisits parallel decomposition algorithms for these types of models. The model is a two-stage stochastic programming problem with first-stage binary variables and second-stage mixed-binary variables. The *here-and-now* decision is to find day-ahead schedules for slow thermal power generators. The *wait-and-see* decision consists of dispatching power and scheduling fast-start generators. We discuss advantages and limitations of different decomposition methods and provide an overview of available software packages. A large-scale numerical example is presented using a modified IEEE 118-bus system with uncertain wind power generation.

## 1 Stochastic Unit Commitment Model

Unit commitment (UC) is a decision-making process that schedules power generation units and production levels over a planning horizon. This process is central to ensuring efficiency and reliability of the power system operation. While fossil-fuel power plants produce 67% of the total electricity generation in the United States (according to 2014 data), renewable power continues to penetrate into the electricity market [21]. This trend has motivated the development of many unit commit-

Kibaek Kim
Mathematics and Computer Science Division
Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439, USA
e-mail: kimk@anl.gov

Victor M. Zavala
Department of Chemical and Biological Engineering, University of Wisconsin-Madison
1415 Engineering Dr, Madison, WI 53706
e-mail: victor.zavala@wisc.edu

First draft: July 17, 2015; Revised: October 28, 2015

ment models that can mitigate uncertainty of renewable supplies. In this chapter, we present a stochastic unit commitment model that determines on-off schedules of the generating units and operation levels, with the objective of cost minimization under uncertain renewable generation. We focus on a day-ahead hourly scheduling of fast and slow generators subject to real-time wind power supply for a 24-hour planning horizon. In this chapter, we use the stochastic unit commitment formulation presented in [14]. We use this model to motivate different algorithmic approaches to solve these types of problems.

We let $\mathscr{T} := \{1,\dots,T\}$ be the set of time periods in the planning horizon. We assume that we have a set $\mathscr{W}$ of wind power generators and that the generation level $W_{jwt}$ from each generator $w \in \mathscr{W}$ is given for each scenario $j \in \mathscr{S}$ and at time $t \in \mathscr{T}$. We also assume that the set $\mathscr{S} := \{1,\dots,S\}$ has only a finite number of scenarios with corresponding probabilities $\{\pi_1,\dots,\pi_S\}$. We let $\mathscr{G}$ be the set of generators considered in the model, and we let $\mathscr{G}_s, \mathscr{G}_f$ be the set of slow generators and fast generators, respectively. We assume that the slow generators are required to be scheduled a day ahead whereas the fast generators can start on demand (i.e., in real-time). Each generator $g \in \mathscr{G}$ starts up, operates, and shuts down at the corresponding costs $C_g^{\text{up}}, C_g^{\text{fx}}$, and $C_g^{\text{dn}}$, respectively. We denote the binary decision variables indicating whether generator $g$ is on or off for each scenario $j$ at time $t$ as $x_{jgt}$. We also denote the binary decision variables indicating whether generator $g$ starts up (or shuts down) for each scenario $j$ at time $t$ by $u_{jgt}$ (or $v_{jgt}$, respectively).

### Logical constraints for commitment, startup and shutdown decisions

The logical relations between commitment, startup, and shutdown decisions are given respectively by

$$1 - x_{jg(t-1)} \geq u_{jgt},\ \forall j \in \mathscr{S}, g \in \mathscr{G}, t \in \mathscr{T}, \tag{1}$$

$$x_{jg(t-1)} \geq v_{jgt},\ \forall j \in \mathscr{S}, g \in \mathscr{G}, t \in \mathscr{T}, \tag{2}$$

$$x_{jgt} - x_{jg(t-1)} = u_{jgt} - v_{jgt},\ \forall j \in \mathscr{S}, g \in \mathscr{G}, t \in \mathscr{T}, \tag{3}$$

where the initial status $x_{jg0}$ of generator $g$ is given for each scenario $j \in \mathscr{S}$. Each generator $g$ has minimum uptime and downtime denoted by $UT_g$ and $DT_g$, respectively. Hence, we have the following constraints:

$$x_{jgt} \geq \sum_{\tau=\max\{1,t-UT_g+1\}}^{t} u_{jg\tau},\ \forall j \in \mathscr{S}, g \in \mathscr{G}, t \in \mathscr{T}, \tag{4}$$

$$1 - x_{jgt} \geq \sum_{\tau=\max\{1,t-DT_g+1\}}^{t} u_{jg\tau},\ \forall j \in \mathscr{S}, g \in \mathscr{G}, t \in \mathscr{T}, \tag{5}$$

with the initial operating status

$$x_{jgt} = 1,\ \forall j \in \mathscr{S}, g \in \mathscr{G}, t \in \{1,\ldots,UT_g^{\text{init}}\}, \tag{6}$$

$$x_{jgt} = 0,\ \forall j \in \mathscr{S}, g \in \mathscr{G}, t \in \{1,\ldots,DT_g^{\text{init}}\}, \tag{7}$$

where $UT_g^{\text{init}}$ and $DT_g^{\text{init}}$ are the initial uptime and downtime of generator $g \in \mathscr{G}$, respectively. The decisions for each slow generator should be the same for all the scenarios, because the slow generators cannot be adjusted in realtime. This is called the *nonanticipativity constraint* and can be expressed mathematically as

$$x_{igt} = x_{jgt},\ u_{igt} = u_{jgt},\ v_{igt} = v_{jgt},\ \forall i, j \in \mathscr{S}, g \in \mathscr{G}_s, t \in \mathscr{T}. \tag{8}$$

**Generation limits, spinning reserve requirements, and ramping constraints**

The model also determines the production levels $p_{jgt}$ and spinning reserve amounts $s_{jgt}$ of generator $g$ at time $t$ for each scenario $j$. The limitations of the generation level are

$$P_g^{\min} x_{jgt} \le p_{jgt} \le P_g^{\max} x_{jgt} - s_{jgt},\ \forall j \in \mathscr{S}, g \in \mathscr{G}, t \in \mathscr{T}, \tag{9}$$

where $P_g^{\min}$ and $P_g^{\max}$ are the minimal and maximal levels of generator $g \in \mathscr{G}$, respectively. The generation rates are physically constrained according to

$$-RD_g \le p_{jgt} - p_{jg(t-1)} \le RU_g - s_{jgt},\ \forall j \in \mathscr{S}, g \in \mathscr{G}, t \in \mathscr{T}, \tag{10}$$

$$s_{jgt} \le RC_g x_{jgt},\ \forall j \in \mathscr{S}, g \in \mathscr{G}, t \in \mathscr{T}, \tag{11}$$

where $RD_g$ and $RU_g$ are the minimal and maximal rates of generation change for each generator $g \in \mathscr{G}$, respectively; $RC_g$ is the ramping capacity of generator $g$; and the initial production level $p_{jg0}$ of generator $g$ is given for each scenario $s$. The constraints (10)-(11) are called *ramping constraints*. The spinning reserve requirement $SR_t$ is given by

$$\sum_{g \in \mathscr{G}} s_{jgt} \ge SR_t,\ \forall j \in \mathscr{S}, t \in \mathscr{T}. \tag{12}$$

**Flow balance and transmission line capacity constraints**

The system balance between the net generation level and load is given by

$$\sum_{g \in \mathscr{G}} p_{jgt} = \sum_{n \in \mathscr{N}} D_{jnt} - \sum_{w \in \mathscr{W}} W_{jwt},\ \forall j \in \mathscr{S}, t \in \mathscr{T}, \tag{13}$$

where $D_{jnt}$ is the demand load of bus $n$ at time $t$ for each scenario $j$. We let $\mathscr{L}$ be the set of transmission lines where each line $l \in \mathscr{L}$ has minimal and maximal capacities denoted by $F_l^{\min}$ and $F_l^{\max}$, respectively. We let $L_{ln}$ be the load shift factor of transmission line $l$ with respect to bus $n$ for each $l \in \mathscr{L}$ and $n \in \mathscr{N}$. The load

shift factor is also known as the power transfer distribution factor that represents the change of power flow on line $l$ with respect to the change in injection at bus $n$. The transmission line capacity is constrained as follows:

$$F_l^{\min} \leq \sum_{g\in\mathscr{G}} L_{ln(g)} p_{jgt} - \sum_{n\in\mathscr{N}} L_{ln} D_{jnt} + \sum_{w\in\mathscr{W}} L_{ln(w)} W_{jwt} \leq F_l^{\max}, \\ \forall j \in \mathscr{S}, l \in \mathscr{L}, t \in \mathscr{T}, \tag{14}$$

where $n(g)$ and $n(w)$ are respectively the indices of buses where generator $g$ and wind farm $w$ are located.

### Piecewise linear objective function

The objective function is the expected cost of operating generators and electricity production. While the cost functions of commitment, startup, and shutdown are linear, the production cost is a nonlinear function that is often approximated by a piecewise linear function. We let $\mathscr{K}$ be the set of linear segments to approximate the cost function. We let $q_{jgkt}$ be the production level of generator $g$ at cost segment $k$ at time $t$ for a given scenario $j$, and we let $Q_{gk}^{\max}$ be the maximum production limit of generator $g$ at cost $C_{gk}^{\mathrm{mar}}$ of segment $k$. The piecewise linear approximation of the cost function is achieved by adding the following constraints:

$$q_{jgkt} \leq Q_{gk}^{\max} x_{jgt}, \ \forall j \in \mathscr{S}, g \in \mathscr{G}, k \in \mathscr{K}, t \in \mathscr{T}, \tag{15}$$

$$p_{jgt} = P_g^{\min} x_{jgt} + \sum_{k\in\mathscr{K}} q_{jgkt}, \ \forall j \in \mathscr{S}, g \in \mathscr{G}, t \in \mathscr{T}. \tag{16}$$

### Stochastic mixed-integer programming formulation

The stochastic unit commitment model is formulated as follows.

$$\min \sum_{j\in\mathscr{S}} \sum_{t\in\mathscr{T}} \sum_{g\in\mathscr{G}} \pi_j \left( C_g^{\mathrm{fx}} x_{jgt} + C_g^{\mathrm{up}} u_{jgt} + C_g^{\mathrm{dn}} v_{jgt} + \sum_{k\in\mathscr{K}} C_{gk}^{\mathrm{mar}} q_{jgkt} \right) \tag{17a}$$

$$\text{s.t. } (1) - (16) \tag{17b}$$

$$x_{jgt} \in \{0,1\}, \ 0 \leq u_{jgt}, v_{jgt} \leq 1, \ p_{jgt}, q_{jgkt}, s_{jgt} \geq 0, \\ \forall j \in \mathscr{S}, g \in \mathscr{G}, k \in \mathscr{K}, t \in \mathscr{T}. \tag{17c}$$

This is a two-stage stochastic mixed-integer program (SMIP) where the first-stage variables represent commitment, startup, and shutdown decisions of slow generators and the second-stage variables represent all the other decisions including those of fast generators for each scenario $j \in \mathscr{S}$. The objective function (17a) is the expected cost of operating generators and producing electricity during the planning horizon $\mathscr{T}$. Note that the integrality restriction is implicitly imposed on variables $u_{jgt}$ and

$v_{jgt}$ for any given binary value $x_{jgt}$. By explicitly relaxing the integrality on $u_{jgt}$ and $v_{jgt}$ the MIP solver can generate tight, valid cutting planes and primal solutions by heuristic procedures.

### Technical challenges of SMIP

SMIP is a challenging problem because the dimensionality increases with the number of scenarios and because the objective function is nonconvex and discontinuous in the first-stage variables. The dimensionality of SMIP makes linear algebra operations in simplex procedures expensive. Moreover, even the linear programming relaxation of the problem might not fit in memory. Consequently, off-the-shelf branch-and-cut solvers are limited to problems with few scenarios. In order to address this issue, different decomposition methods have been proposed. Benders decomposition (also known as the L-shaped method) has been the most popular method for solving SMIPs, but this algorithm does not have convergence guarantees when integer variables are present in the second stage (e.g., startup and shutdown of fast generators). Therefore, other approaches are required. These include specialized branch-and-bound techniques using tender variables [3] and various convexification techniques (e.g., Gomory cuts [9, 24], mixed-integer rounding cuts [13], and disjunctive cuts [19, 20]). However, they are limited to certain problem classes. Consequently, we are interested in decomposition methods that can solve SMIP to optimality or at least provide good upper and/or lower bounds. This can be achieved by using a variety of methods such as dual decomposition and progressive hedging. Dual decomposition is implemented in the open-source package `DSP` [14], and progressive hedging is implemented in `PySP` [23]. `DSP` provides a `Julia`-based modeling interface (i.e., `StochJuMP` [12]), and `PySP` provides a `Python`-based modeling interface. The `ddsip` package [17] also implements a dual decomposition method but does not support model specification through a standard file format (i.e., SMPS [5]) and algebraic modeling languages.

## 2 Scenario Decomposition

In this section, we present scenario decomposition methods for the stochastic unit commitment model presented in Section 1. For simplicity, we write this model in the general form

$$z = \min_{x_j, y_j} \quad \sum_{j \in \mathscr{S}} \pi_j \left( c^T x_j + q_j^T y_j \right) \tag{18a}$$

$$\text{s.t.} \quad \sum_{j \in \mathscr{S}} H_j x_j = 0, \quad (\lambda) \tag{18b}$$

$$(x_j, y_j) \in G_j, \ \forall j \in \mathscr{S}, \tag{18c}$$

where $x$ is the first-stage decision vector and $y_j$ is the second-stage decision vector for each scenario $j \in \mathscr{S}$. Equation (18b) represents the nonanticipativity constraints (8) with a suitable matrix $H_j \in \mathbb{R}^{S \cdot n_1 \times n_1}$, and $\lambda \in \mathbb{R}^{S \cdot n_1}$ is the corresponding dual variable. In equation (18c), the set $G_j$ of feasible solutions is defined by all the constraints except the nonanticipativity constraints.

## 2.1 Dual Decomposition

Dual decomposition was first proposed in [6]. This method applies a Lagrangian relaxation of the nonanticipativity constraints (8) to obtain the Lagrangian dual function

$$D(\lambda) := \min_{x_j, y_j} \left\{ \sum_{j \in \mathscr{S}} L_j(x_j, y_j, \lambda) : (x_j, y_j) \in G_j, \ \forall j \in \mathscr{S} \right\}, \tag{19}$$

where

$$L_j(x_j, y_j, \lambda) := \pi_j \left( c^T x_j + q_j^T y_j \right) + \lambda^T (H_j x_j). \tag{20}$$

For fixed $\lambda$, the Lagrangian dual function can be decomposed as

$$D(\lambda) = \sum_{j \in \mathscr{S}} D_j(\lambda), \tag{21}$$

where

$$D_j(\lambda) := \min_{x_j, y_j} \left\{ L_j(x_j, y_j, \lambda) : (x_j, y_j) \in G_j \right\}. \tag{22}$$

We thus seek to obtain the best lower bound for (18) by solving the Lagrangian dual problem:

$$z_{\mathrm{LD}} := \max_{\lambda} \sum_{j \in \mathscr{S}} D_j(\lambda). \tag{23}$$

Proposition 1 is an important property of the Lagrangian relaxation, which shows the tightness of the lower bound $z_{\mathrm{LD}}$ [10].

**Proposition 1.** *The optimal value $z_{LD}$ of the Lagrangian dual problem (23) is equal to the optimal value of the following linear program,*

$$\min_{x_j, y_j} \left\{ \sum_{j \in \mathscr{S}} \pi_j \left( c^T x_j + q_j^T y_j \right) : \sum_{j \in \mathscr{S}} H_j x_j = 0, \ (x_j, y_j) \in \mathit{conv}(G_j), \ \forall j \in \mathscr{S} \right\}, \tag{24}$$

*where $\mathit{conv}(G_j)$ denotes the convex hull of $G_j$. Moreover, $z_{LD} \geq z_{LP}$ holds, where $z_{LP}$ is the optimal value of the linear programming relaxation of (18).*

### 2.1.1 Subgradient Method

Subgradient methods have been widely used in nonsmooth optimization. We let $\lambda^k$ be the dual variable at iteration $k \geq 0$, and we let $x_s^k$ be an optimal solution of (22) for given $\lambda^k$. The dual variable is updated as

$$\lambda^{k+1} = \lambda^k - \alpha_k \sum_{j \in \mathscr{S}} H_j x_j^k, \tag{25}$$

where $\alpha_k \in (0,1]$ is the step size. This method updates the duals by using a subgradient of $D(\lambda)$ at $\lambda^k$, denoted by $\sum_{j \in \mathscr{S}} H_j x_j^k$. Different step-size rules have been studied for subgradient methods [4]. For example, in [7] the step size $\alpha_k$ is given by

$$\alpha_k := \beta_k \frac{z_{\mathrm{UB}} - D(\lambda^k)}{\left\| \sum_{j \in \mathscr{S}} H_j x_j^k \right\|_2^2}, \tag{26}$$

where $z_{\mathrm{UB}}$ is the objective value of the best-known feasible solution to (18) up to iteration $k$ and $\beta_k$ is a user-defined positive scalar. The subgradient method is summarized in Algorithm 1.

**Algorithm 1** Dual Decomposition Based on Subgradient Method (DD-Sub)

1: Set $k \leftarrow 0, z_{\mathrm{LB}} \leftarrow -\infty, z_{\mathrm{UB}} \leftarrow \infty$ and $\gamma \leftarrow 0$.
2: **loop**
3: SOLVE (22) to obtain $D_j(\lambda^k)$ and $(x_j^k, y_j^k)$ for given $\lambda^k$ and for all $j \in \mathscr{S}$
4: **if** $D(\lambda^k) > z_{\mathrm{LB}}$ **then**
5: $z_{\mathrm{LB}} \leftarrow D(\lambda^k)$
6: **else**
7: $\gamma \leftarrow \gamma + 1$
8: **if** $\gamma = \gamma^{\max}$ **then**
9: $\beta_k \leftarrow 0.5\beta_k$ and $\gamma \leftarrow 0$
10: **end if**
11: **end if**
12: UPDATE $z_{\mathrm{UB}}$ for given $x_s^k$
13: $k \leftarrow k + 1$
14: **end loop**

Algorithm 1 is initialized with user-defined parameters $\lambda^0, \gamma^{\max}$, and $\beta_0$ and reduces $\beta_k$ by half when the best lower bound $z_{\mathrm{LB}}$ is not improved for the last $\gamma^{\max}$ iterations (lines 8-10). The best upper bound $z_{\mathrm{UB}}$ may be obtained by solving (18) for fixed $x_s^k$ (line 12). An important limitation of the subgradient method is that finite termination cannot be proved [7].

### 2.1.2 Cutting-Plane Method

The cutting-plane method is an outer approximation scheme that solves the Lagrangian dual problem by iteratively adding linear inequalities. The outer approximation of (23) at iteration $k$ is given by the Lagrangian master problem:

$$m_k := \max_{\theta_j, \lambda} \sum_{j \in \mathscr{S}} \theta_j \tag{27a}$$

$$\text{s.t. } \theta_j \leq D_j(\lambda^l) + \left(H_j x_j^l\right)^T (\lambda - \lambda^l),\ \forall j \in \mathscr{S},\ l = 0, 1, \ldots, k. \tag{27b}$$

The dual variable $\lambda^{k+1}$ is obtained by solving the approximation (27) at iteration $k$. We define the primal-dual solution of the Lagrangian master problem as the triplet $(\theta, \lambda, \pi)$. Here, $\theta := (\theta_1, ..., \theta_S)$ and $\pi := (\pi_1^0, ..., \pi_1^k, ..., \pi_S^0, ..., \pi_S^k)$, where $\pi_s^l$ are the dual variables of (27b). The master problem (27) exhibits a dual block-angular structure that allows for parallelism in solving the master by using an interior-point solver [16]. The method is summarized in Algorithm 2.

**Algorithm 2** Dual Decomposition Based on Cutting-Plane Method

1: $k \leftarrow 0$ and $\lambda^0 \leftarrow 0$
2: SOLVE (22) to obtain $D_j(\lambda^k)$ and $(x_j^k, y_j^k)$ for given $\lambda^k$ and for each $j \in \mathscr{S}$
3: $z_{\text{LB}} \leftarrow D(\lambda^k)$.
4: **repeat**
5: ADD (27b) for given $D(\lambda^k)$ and $x_j^k$
6: SOLVE (27) to obtain $m_k$ and $(\theta^{k+1}, \lambda^{k+1})$
7: SOLVE (22) to obtain $D_j(\lambda^{k+1})$ and $(x_j^{k+1}, y_j^{k+1})$ for given $\lambda^{k+1}$ and for all $j \in \mathscr{S}$
8: UPDATE $z_{\text{LB}} \leftarrow \max\{z_{\text{LB}}, D(\lambda^{k+1})\}$.
9: $k \leftarrow k + 1$
10: **until** $m_{k-1} \leq D(\lambda^k)$

The function $D_j(\lambda)$ is piecewise linear concave in $\lambda$ supported by the linear inequalities (27b). Assuming that the master problem (27) and the subproblem (22) can be solved to optimality, Algorithm 2 terminates with an *optimal* solution of (23) after a finite number of steps because the number of linear inequalities required to approximate $D(\lambda)$ is finite. This gives the cutting-plame method a natural termination criterion (i.e., $m_{k-1} \leq D(\lambda^k)$). In other words, this criterion indicates that $m_{k-1}$ matches the Lagrangian dual function $D(\lambda^k)$ and thus the maximum of the Lagrangian master problem matches the maximum of the Lagrangian dual problem.

### 2.1.3 Variants of the Cutting-Plane Method

The cutting-plane method is inherently unstable, and hence the solutions of the master problem (27) oscillate significantly when the Lagrangian dual (23) is not well approximated at the beginning of the iterations. Moreover, the solution also suffers

from degeneracy. Several variants of the cutting-plane method have been proposed to overcome these issues. Here we present two variants.

### Interior-Point Cutting-Plane Method

The interior-point method (IPM) with early termination criteria was proposed in [14]. The IPM solves the master problem (27) suboptimally to find stronger cuts from interior feasible solutions and to avoid degeneracy.

The IPM checks the termination criteria in the following order:

1. $\sum_{j\in\mathscr{S}} \theta_j^k \geq z_{\mathrm{UB}}$
2. $g_k(\theta^k,\lambda^k,\mu^k) < \varepsilon_{\mathrm{IPM}}^k$

Here, $g_k(\theta^k,\lambda^k,\mu^k)$ is the relative duality gap of the primal-dual feasible solution $(\theta^k,\lambda^k,\mu^k)$ of the master (27) at iteration $k$. The tolerance $\varepsilon_{\mathrm{IPM}}^k$ can be relaxed when the duality gap of the dual decomposition method is large. It is updated as follows:

**Algorithm 3** Dual Decomposition Based on Interior-Point Cutting-Plane Method (IPCPM)

$k \leftarrow 0, \lambda^0 \leftarrow 0$ and $z_{\mathrm{UB}} \leftarrow \infty$
SOLVE (22) to obtain $D_j(\lambda^k)$ and $(x_j^k, y_j^k)$ for given $\lambda^k$ and for each $j \in \mathscr{S}$
ADD cutting-planes (27b) to the master (27) for given $D(\lambda^k)$ and $x_j^k$
$z_{\mathrm{LB}} \leftarrow D(\lambda^k)$.
**loop**
  SOLVE the master (27) by the IPM to obtain $(\theta^{k+1},\lambda^{k+1})$
  SOLVE (22) to obtain $D_j(\lambda^{k+1})$ and $(x_j^{k+1}, y_j^{k+1})$ for given $\lambda^{k+1}$ and for each $j \in \mathscr{S}$
  **if** $(\theta^{k+1},\lambda^{k+1})$ is obtained from the first termination criterion **then**
    **if** $\theta_j^{k+1} \leq D_j(\lambda^{k+1})$ for all $j \in \mathscr{S}$ **then**
      STOP
    **else**
      ADD cutting-planes (27b) to the master (27) for given $D(\lambda^{k+1})$ and $x_s^{k+1}$
    **end if**
  **else if** $(\theta^{k+1},\lambda^{k+1})$ is obtained from the second termination criterion **then**
    **if** $\theta_j^{k+1} \leq D_j(\lambda^{k+1})$ for all $j \in \mathscr{S}$ **then**
      **if** $\varepsilon_{\mathrm{IPM}}^k > \varepsilon_{\mathrm{Opt}}$ **then**
        UPDATE $\varepsilon_{\mathrm{IPM}}^{k+1}$ from (28)
      **else**
        STOP
      **end if**
    **else**
      ADD cutting-planes (27b) to the master (27) for given $D(\lambda^{k+1})$ and $x_j^{k+1}$
    **end if**
  **end if**
  $z_{\mathrm{LB}} \leftarrow \max\{z_{\mathrm{LB}}, D(\lambda^k)\}$.
  $k \leftarrow k+1$
**end loop**

$$\varepsilon_{\text{IPM}}^{k} := \min\left\{\varepsilon_{\text{IPM}}^{max}, \frac{g_{k-1}(\tilde{\theta}^{k-1}, \tilde{\pi}^{k-1})}{\delta} + \frac{\tilde{m}_{k-1} - \sum_{j\in\mathscr{S}} D_j(\tilde{\lambda}^{k-1})}{1+|\tilde{m}_{k-1}|}\right\}, \tag{28}$$

where $\tilde{m}_{k-1} := \sum_{j\in\mathscr{S}} \tilde{\theta}_j^{k-1}$ and $\delta > 1$ is the *degree of optimality* [11]. The interior-point cutting-plane method is summarized in Algorithm 3. Algorithm 3 terminates after a finite number of iterations with an optimal solution of the Lagrangian dual problem (23) (see Theorem 2 in [14] for a proof).

### Bundle Method

The bundle method is a stabilized cutting-plane method where a quadratic stabilizing term is added to the objective function of the master (27). As a result, the method solves the master problem

$$\max_{\theta_j,\lambda} \sum_{j\in\mathscr{S}} \theta_j + \frac{1}{2\tau}\|\lambda - \lambda^{+}\|^2 \tag{29a}$$

$$\text{s.t. } \theta_j \le D_j(\lambda^l) + \left(H_j x_j^l\right)^T (\lambda - \lambda^l),\ \forall j \in \mathscr{S},\ l = 0,1,\ldots,k, \tag{29b}$$

where $\lambda^+$ is a *stability center* and $\tau > 0$ is a parameter that defines the *stabilization effect* [15]. Note that the bundle method becomes equivalent to the cutting-plane method for a large $\tau$ and the subgradient method for a small $\tau$. A bundle method for dual decomposition is summarized in Algorithm 4 as presented in [16], where

**Algorithm 4** Dual Decomposition Based on Bundle Method

Choose initial $\kappa \in (0,1)$ and $t > 0$, $k \leftarrow 0, \lambda^+ \leftarrow 0$ and $\lambda^0 \leftarrow 0$
SOLVE (22) to obtain $D_j(\lambda^k)$ and $(x_j^k, y_j^k)$ for given $\lambda^k$ and for each $j \in \mathscr{S}$
$z_{\text{LB}} \leftarrow D(\lambda^k)$
**loop**
  ADD cuts (29b) to the master (29) for given $D(\lambda^k)$ and $x_j^k$
  SOLVE the master (29) to obtain $(\theta^{k+1}, \lambda^{k+1})$
  $v \leftarrow \sum_{j\in\mathscr{S}} \theta_j^{k+1} - z_{\text{LB}}$
  **if** $v < \varepsilon(1+|z_{\text{LB}}|)$ **then**
    STOP
  **end if**
  $k \leftarrow k+1$
  SOLVE (22) to obtain $D_j(\lambda^k)$ and $(x_j^k, y_j^k)$ for given $\lambda^k$ and for each $j \in \mathscr{S}$.
  $u \leftarrow 2t\left[1 - \left(D(\lambda^k) - z_{\text{LB}}\right)/v\right]$
  UPDATE $t \leftarrow \min\left\{\max\left\{u, t/10, 10^{-4}\right\}, 10t\right\}$
  **if** $D(\lambda^k) - D(\lambda^+) > \kappa v$ **then**
    UPDATE $z_{\text{LB}} \leftarrow D(\lambda^k)$
    UPDATE $\lambda^+ \leftarrow \lambda^k$
  **end if**
**end loop**

the dual of the master problem (27) is solved. Algorithm 4 is convergent in the limit [16].

We also note that stabilization of the master solution can be achieved by using a trust-region constraint (as opposed to the quadratic term used in the bundle method), as is done in [14]. The trust region avoids the need of tuning the parameter $\tau$.

## 2.2 Progressive Hedging

Progressive hedging is a scenario decomposition framework that is partly motivated by the augmented Lagrangian dual of (18)

$$\begin{aligned} L_r(\hat{x},\lambda) &= \sum_{j\in\mathscr{S}} \pi_j \left(c^T x_j + q_j^T y_j\right) + \sum_{j\in\mathscr{S}} \lambda^T \left(H_j x_j\right) + 0.5\rho \| \sum_{j\in\mathscr{S}} H_j x_j\|^2 \\ &= \sum_{j\in\mathscr{S}} \pi_j \left(c^T x_j + q_j^T y_j\right) + \sum_{j\in\mathscr{S}} \omega_j^T x_j + 0.5\rho \| \sum_{j\in\mathscr{S}} \pi_j x_j - \hat{x}\|^2, \end{aligned} \tag{30}$$

where $\omega_j := \lambda^T H_j$, and the second equality holds because the nonanticipativity constraints $\sum_{j\in\mathscr{S}} H_j x_j = 0$ is equivalent to $\sum_{j\in\mathscr{S}} \pi_j x_j - \mathbb{E}[x] = 0$ [18]. An important observation is that the term $\|\sum_{j\in\mathscr{S}} \pi_j x_j - \hat{x}\|^2$ of (30) cannot be decomposed in scenarios. To achieve decomposition, the progressive hedging algorithm solves subproblems of the form

$$P_j(\hat{x},\omega,\rho) := \min\left\{\pi_j \left(c^T x_j + q_j^T y_j\right) + \omega^T x_j + 0.5\|\rho^T (x_j - \hat{x})\|^2 : (x_j, y_j) \in G_j\right\}, \tag{31}$$

where the vector $\hat{x}$ is the expected value of $x$ from the previous iteration, $\omega \in \mathbb{R}^{n_1}$ is a price vector and $\rho > 0$ is a perturbation vector. For computational efficiency, the quadratic proximal term of (31) is often approximated by using piecewise linear functions [23]. We note that the progressive hedging subproblem (31) has structural connections with the augmented Lagrangian (30). Unlike the (augmented) Lagrangian relaxation, however, the progressive hedging method seeks to find a feasible solution (i.e., an upper bound). The method is summarized in Algorithm 5.

The choice of the perturbation vector $\rho$ can significantly affect algorithmic performance. A small value of $\rho$ may require many iterations to achieve changes in the first-stage variables, whereas a large value may lead to a suboptimal solution. Different strategies for computing $\rho$ are discussed in [22]. We also note that Algorithm 5 cannot be guaranteed to terminate in a finite number of steps, but it often provides good-quality solutions.

**Algorithm 5** Progressive Hedging (PH)

1: $k \leftarrow 0$ and $\omega_j^0 \leftarrow 0$ for each $j \in \mathscr{S}$
2: SOLVE $P_j(0,0,0)$ to obtain $(x_j^k, y_j^k)$ for all $j \in \mathscr{S}$
3: $z_{\text{LB}} \leftarrow \sum_{j\in\mathscr{S}} P_j(0,0,0)$.
4: $k \leftarrow k+1$
5: **repeat**
6: UPDATE $\hat{x}^{k-1} \leftarrow \sum_{j\in\mathscr{S}} \pi_j x_j^{k-1}$
7: UPDATE $\omega_j^k \leftarrow \omega_j^k + \rho\left(x_j^{k-1} - \hat{x}_j^{k-1}\right)$ for each $j \in \mathscr{S}$.
8: SOLVE $P_j(\hat{x}_j^{k-1}, \omega_j^k, \rho)$ to obtain $(x^k, x^k)$ for each $j \in \mathscr{S}$.
9: $k \leftarrow k+1$
10: **until** $x_j^k$ are equal for all $j \in \mathscr{S}$

## *2.3 Incorporating Benders-type Cutting-Plane Procedure*

One can combine dual decomposition and Benders techniques. For instance, a Benders-type cutting-plane procedure has been proposed for the Lagrangian subproblems (22) in [14]. The aim of the cutting-plane procedure is to eliminate infeasible first-stage solutions and to tighten the Lagrangian subproblems. The computational experiments in [14] show significant improvement in the quality of the bounds, number of iterations, and solution time. For simplicity, we use the general definition of set $G_j := \{(x_j, y_j) : Ax_j = b,\ T_j x_j + W_j y_j = h_j\}$ for each $j \in \mathscr{S}$.

### Feasibility Cuts

Without loss of generality, we let $(\hat{x}_1, \hat{y}_1) \in G_1$ be the feasible subproblem solution that is infeasible to the original problem (18). This situation can occur because the stochastic UC problem might not have relatively complete recourse. For $j \in \mathscr{S}$, we solve the linear program

$$\max_{\mu} \left\{ \mu^T (h_j - T_j \hat{x}_1) : \mu^T W_j \le 0,\ |\mu| \le 1 \right\}, \tag{32}$$

where the absolute value $|\cdot|$ is taken componentwise, until $\mu_j^T (h_j - T_j x) > 0$ for some $j$. For some $j$ such that $\mu_j^T (h_j - T_j x) > 0$, we add

$$\mu_j^T (h_j - T_j x) \le 0 \tag{33}$$

to the constraint set of $G_j$ for all $j \in \mathscr{S}$.

This cut eliminates a candidate first-stage solution $\hat{x}_1$ that does not have a feasible recourse for the subproblem. We highlight, however, that infeasibility can be guaranteed to be eliminated only if it is detected at the root node of the branch-and-bound tree of the subproblem (22).

Optimality Cuts

We assume that $(\hat{x}_1, \hat{y}_1) \in G_1$ is a feasible subproblem solution that is also feasible with respect to the original problem (18). For $j \in \mathscr{S}$, we solve the linear program

$$\max_{\pi} \left\{ \pi^T \left( h_j - T_j \hat{x}_1 \right) : \pi^T W_j \leq q_j \right\}. \tag{34}$$

The optimality cut is generated by

$$c^T x + \sum_{j \in \mathscr{S}} \pi_j^T \left( h_j - T_j x \right) \leq z_{\text{UB}} \tag{35}$$

for a given best upper bound $z_{\text{UB}}$ and added to the constraint set of $G_j$ for all $j \in \mathscr{S}$.

We note that the optimality cut (35) is parameterized by the best-known upper bound $z_{\text{UB}}$ and thus can be tightened as better upper bounds are obtained. In other words, the optimality cut seeks to eliminate first-stage solutions that go above a known upper bound.

Procedure 1 summarizes the Benders-type cutting-plane procedure for solving the Lagrangian subproblems (22) by adding the valid inequalities (33) and (35). The procedure can also be applied to other scenario decomposition methods. Moreover, the procedure terminates in a finite number of steps (see Theorem 1 in [14]).

**Procedure 1** Cutting-Plane Procedure for Lagrangian Subproblems (CPSub)

**Require:** $\lambda^k$
**for all** $s \in \mathscr{S}$ **do**
  **repeat**
    SOLVE subproblem (22) to obtain $D_j(\lambda^k)$ and $(x_j^k, y_j^k)$ for $\lambda^k$
    $isFeasible \leftarrow true$
    **for all** $j' \in \mathscr{S} \backslash \{j\}$ **do**
      SOLVE feasibility cut generator (32) to obtain $\mu_{j'}$ for $x_j^k$
      **if** $\mu_{j'}^T (h_{j'} - T_{j'} x_j^k) > 0$ **then**
        ADD feasibility cut (33) to all the subproblems (22)
        $isFeasible \leftarrow false$
      **end if**
    **end for**
  **until** $isFeasible = true$
  UPDATE $z_{\text{UB}}$ by solving (18) for fixed $x_j^k$
  GENERATE optimality cut (35) by solving (34) for $x_j^k$ and for all $j \in \mathscr{S}$
  ADD optimality cut (35) to all the subproblems (22)
**end for**

## 3 Numerical Example

We present a numerical example of the stochastic unit commitment model. We also illustrate how to solve the model by using open-source software packages. In this model, thermal power generators are scheduled over a day. The schedules are subject to uncertainty in wind power generation. We use a modified IEEE 118-bus system with 54 generators, 118 buses, and 186 transmission lines provided in [14]. We assume that 17 of the 54 generators are allowed to start on demand (second stage) whereas the other generators should be scheduled in advance (first stage). We also consider 3 identical wind farms, each consisting of 120 wind turbines. The demand load is 3,095 MW on average, with a peak of 3,733 MW. The wind power generation level is 494 MW on average, with a peak of 916 MW for the 64 scenarios generated. Figure 1 shows the 64 scenarios (grey lines) of wind power generation and the mean levels (red lines). We used real wind-speed data predicted from the observations of 31 weather stations in the state of Illinois.

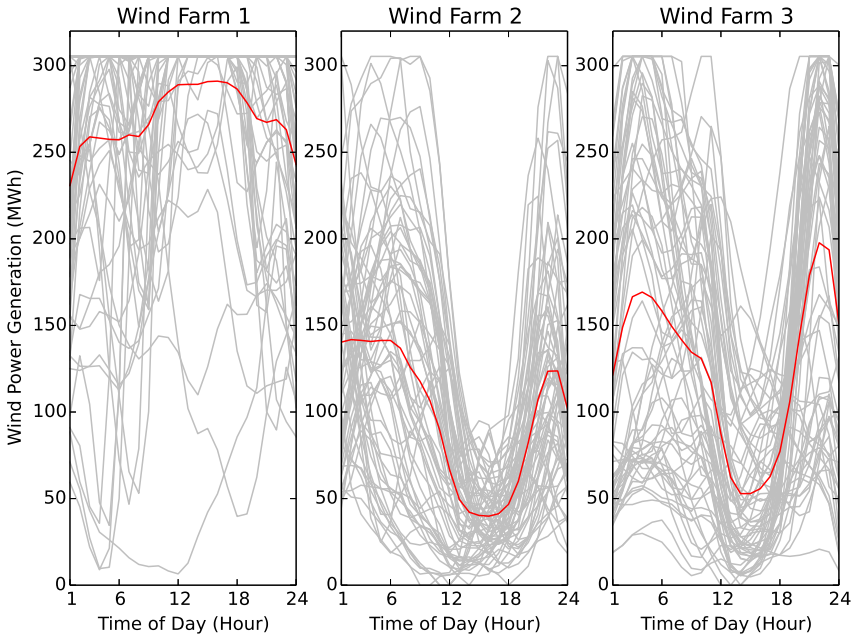


**Fig. 1** Wind power generation scenarios for wind farms considered in the stochastic unit commitment model.

Table 1 presents the size of the stochastic unit commitment instances with 4, 8, 16, 32, and 64 scenarios. The first stage has 10,727 constraints and 2,592 variables

**Table 1** Characteristics of stochastic unit commitment instances

| # Scenarios | # Rows | # Columns | # Integers |
|---|---|---|---|
| 4 | 120,015 | 38,880 | 2,592 |
| 8 | 229,303 | 75,168 | 4,320 |
| 16 | 447,879 | 147,744 | 7,776 |
| 32 | 885,031 | 292,896 | 14,688 |
| 64 | 1,759,335 | 583,200 | 28,512 |

including 864 integer variables, and the second stage has 27,322 constraints and 9,072 variables including 432 integer variables.

We also solved the extensive form of the problems by using `SCIP` [2] (version 3.1.1) with a 6-hour time limit and a 0.01% optimality gap tolerance. Table 2 presents the computational results for the extensive form. None of the instances except the 4-scenario instance can reach the 0.01% optimality gap within the 6-hour limit. Moreover, the 64-scenario instance finds a poor lower bound without an upper bound.

To avoid scalability issues, we solved the stochastic UC problems by using the scenario decomposition methods presented in Section 2. Specifically, we used the `DSP` package [14] (version 0.2.0) and `PySP` [23] in the `Pyomo` package (version 4.0.9682) for the dual decomposition and progressive hedging methods, respectively. `DSP` and `PySP` solve the subproblems with 0.01% of optimality gap by using `SCIP` (version 3.1.1) and `CBC` [8] (version 2.8), respectively. We could not use `SCIP` for `PySP` because the interface of `SCIP` for `PySP` does not allow us to set the optimality gap tolerance. All computations were run on *Blues*, a 310-node computing cluster at Argonne National Laboratory. Each node on the Blues cluster has two 8-core 2.6 GHz Xeon processors and 64 GB of RAM.

**Table 2** Numerical results for the extensive form of the stochastic unit commitment problems

| # Scenarios | Branch-and-Cut Nodes | Upper Bound | Lower Bound | Gap (%) | Time (sec.) |
|---|---|---|---|---|---|
| 4 | 88831 | 907035.3 | 906089.9 | 0.01 | 6632 |
| 8 | 58235 | 904068.1 | 903567.8 | 0.05 | $>21600$ |
| 16 | 3505 | 900806.1 | 900200.3 | 0.07 | $>21600$ |
| 32 | 9 | 907536.0 | 901759.8 | 0.64 | $>21600$ |
| 64 | 1 | $\infty$ | 33605.4 | $\infty$ | $>21600$ |

## *3.1 Lower and Upper Bounds from Dual Decomposition*

We first solved the problems using the dual decomposition method implemented in the `DSP` solver. We used the interior-point cutting-plane method (IPCPM) for solving the master problem (27). We also used the Benders-type cutting-plane procedure (CPSub). `DSP` is capable of solving a problem in parallel with up to $S^2$ number of

computing cores (i.e., up to 4,096 cores for the 64-scenario instance). However, in our numerical example, we used only up to 2,048 cores on the Blues cluster.

**Table 3** Numerical results for stochastic unit commitment problem from `DSP`.

| # Scenarios | Iterations | Upper Bound | Lower Bound | Gap (%) | Time/Iter. (sec.) | Total Time (sec.) |
|---|---|---|---|---|---|---|
| 4 | 1 | 907046.1 | 906979.1 | $< 0.01$ | 551 | 551 |
| 8 | 1 | 904006.6 | 903953.5 | $< 0.01$ | 667 | 667 |
| 16 | 1 | 900706.3 | 900650.7 | $< 0.01$ | 764 | 764 |
| 32 | 19 | 903227.7 | 903149.9 | $< 0.01$ | 390 | 7424 |
| 64 | 16 | 895118.0 | 895044.6 | $< 0.01$ | 895 | 14320 |

Table 3 shows that upper and lower bounds are obtained with $< 0.01\%$ optimality gap for all the problem instances. Moreover, the `DSP` solutions terminate after the first iteration for the 4-, 8- and 16-scenario instances because the CPSub procedure excluded infeasible first-stage solutions and tightened the scenario subproblems. We note that the solution time per iteration does not increase with the number of scenarios because of parallelization of `DSP`.

We present results for the different variants of dual decomposition discussed in Section 2. We compare: the subgradient method (DDSub), IPCPM *without* CPSub

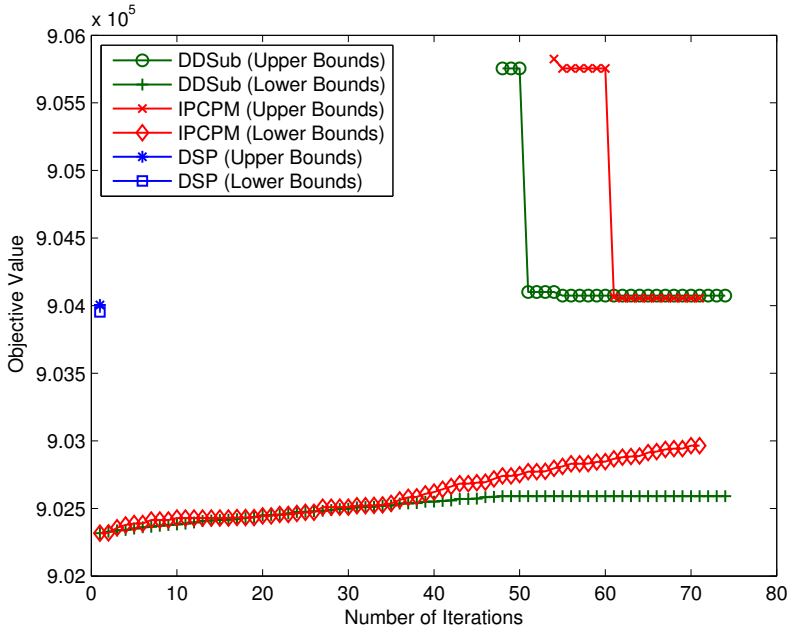


**Fig. 2** Upper bounds and lower bounds obtained with DSP with and without Procedure 1 and with the subgradient method.

(IPCPM), and IPCPM *with* CPSub (DSP). Figure 2 shows the best upper bound and the best lower bound obtained at each iteration. As can be seen, DSP obtains upper and lower bounds with $< 0.01\%$ duality gap at the first iteration and achieved termination, whereas IPCPM and DDSub are not able to find upper bounds for the first 53 iterations and the first 47 iterations, respectively. The results clearly indicate that *the problem does not have relatively complete recourse*. Moreover, DSP finds tighter lower bounds than do DDSub and IPCPM because of the ability to tighten the subproblems by Procedure 1. The figure also shows that IPCPM finds better lower and upper bounds than does DDSub.

### *3.2 Upper Bounds from Progressive Hedging*

We now present results for the progressive hedging method implemented in the `PySP` package. The subproblem solutions are parallelized by using the `Pyro` package [1] that provides capabilities for distributed computing. In our numerical example, we set the initial perturbation vector $\rho := 1.0$ and later adjust it in proportion to the objective function coefficient [23]. We also set the `--enable-ww-extension` option, which enables additional heuristics for finding feasible solutions [22]. We note that one could tune the parameters and devise other heuristics.

Table 4 presents the results from solving the stochastic UC problems by the progressive hedging method in `PySP`. `PySP` finds tight upper bounds for the 4-, 8, 16-, and 32-scenario problems. For the 64-scenario problem, however, `PySP` cannot find an upper bound after 53 iterations and 6 hours of solution time. We also note that parallelization keeps the time per iteration constant as we increase the number of scenarios. The number of iterations, however, also tends to increase as we increase the number of scenarios, because the number of nonanticipativity constraints increases.

**Table 4** Numerical results for stochastic unit commitment problem from `PySP`.

| # Scenarios | Iterations | Upper Bound | Time/Iter. (sec.) | Total Time (sec.) |
|---|---|---|---|---|
| 4 | 5 | 907042.5 | 245 | 1224 |
| 8 | 13 | 904041.2 | 203 | 2645 |
| 16 | 26 | 900712.1 | 240 | 6228 |
| 32 | 20 | 903355.7 | 603 | 12069 |
| 64 | 53 | $\infty$ | 407 | $> 21600$ |

## 4 Summary

We have presented a stochastic unit commitment model, which can be formulated as a two-stage stochastic mixed-integer programming problem. The first stage sched-

ules slow generation units a day ahead under uncertain wind power production The second stage dispatches electricity production in order to meet demand load, while scheduling fast generation units. The problem is challenging because it is a large-scale mixed-integer programming problem. Moreover, standard Benders decomposition cannot be applied because the recourse function is nonconvex and discontinuous on the first-stage variables. As a result, we have presented scenario decomposition methods, which split the original problem into the scenario number of subproblems and solve the smaller subproblems. This decomposability naturally leads to parallel algorithms capable of running on high-performance computing systems. We have presented dual decomposition and progressive hedging with various enhancement techniques that incorporate Benders-type cuts in the scenario decomposition framework. In the numerical example, we used an IEEE 118-bus system with 64 scenarios of wind power productions. We solved the stochastic unit commitment problem using open-source software packages, `DSP` and `PySP`, that implement parallel dual decomposition and progressive hedging, respectively. The numerical results show that state-of-the-art solution methods are able to solve a large-scale stochastic unit commitment problem. However, more research is needed to scale up these methods and address a large number of scenarios.

**Acknowledgements** This material is based upon work supported by the U.S. Department of Energy, Office of Science, under contract number DE-AC02-06CH11357. We gratefully acknowledge the computing resources provided on Blues, a high-performance computing cluster operated by the Laboratory Computing Resource Center at Argonne National Laboratory. We thank Julie Bessac for providing wind speed prediction data.

## References

1. Pyro: Python remote objects (2015). URL http://pythonhosted.org/Pyro4/
2. Achterberg, T.: Scip: solving constraint integer programs. Mathematical Programming Computation **1**(1), 1–41 (2009)
3. Ahmed, S., Tawarmalani, M., Sahinidis, N.V.: A finite branch-and-bound algorithm for two-stage stochastic integer programs. Mathematical Programming **100**(2), 355–377 (2004)
4. Bertsekas, D.P., Scientific, A.: Convex Optimization Algorithms. Athena Scientific (2015)
5. Birge, J.R., Dempster, M.A., Gassmann, H.I., Gunn, E.A., King, A.J., Wallace, S.W.: A standard input format for multiperiod stochastic linear programs. IIASA Laxenburg Austria (1987)
6. CarøE, C.C., Schultz, R.: Dual decomposition in stochastic integer programming. Operations Research Letters **24**(1), 37–45 (1999)
7. Fisher, M.L.: An applications oriented guide to Lagrangian relaxation. Interfaces **15**(2), 10–21 (1985)
8. Forrest, J.: Cbc. URL https://projects.coin-or.org/Cbc
9. Gade, D., Küçükyavuz, S., Sen, S.: Decomposition algorithms with parametric gomory cuts for two-stage stochastic integer programs. Mathematical Programming **144**(1-2), 39–64 (2014)
10. Geoffrion, A.M.: Lagrangean relaxation for integer programming. Springer (1974)
11. Gondzio, J., Gonzalez-Brevis, P., Munari, P.: New developments in the primal–dual column generation technique. European Journal of Operational Research **224**(1), 41–51 (2013)

12. Huchette, J., Lubin, M., Petra, C.: Parallel algebraic modeling for stochastic optimization. In: Proceedings of the 1st First Workshop for High Performance Technical Computing in Dynamic Languages, pp. 29–35. IEEE Press (2014)
13. Kim, K., Mehrotra, S.: A two-stage stochastic integer programming approach to integrated staffing and scheduling with application to nurse management. Optimization Online (2014)
14. Kim, K., Zavala, V.M.: Algorithmic innovations and software for the dual decomposition method applied to stochastic mixed-integer programs. Optimization Online (2015)
15. Lemaréchal, C.: Lagrangian relaxation. In: Computational combinatorial optimization, pp. 112–156. Springer (2001)
16. Lubin, M., Martin, K., Petra, C.G., Sandıkçı, B.: On parallelizing dual decomposition in stochastic integer programming. Operations Research Letters **41**(3), 252–258 (2013)
17. Märkert, A., Gollmer, R.: Users Guide to ddsip–A C package for the dual decomposition of two-stage stochastic programs with mixed-integer recourse (2014)
18. Rockafellar, R.T., Wets, R.J.B.: Scenarios and policy aggregation in optimization under uncertainty. Mathematics of operations research **16**(1), 119–147 (1991)
19. Sen, S., Higle, J.L.: The $C^3$ theorem and a $D^2$ algorithm for large scale stochastic mixed-integer programming: set convexification. Mathematical Programming **104**(1), 1–20 (2005)
20. Sherali, H.D., Fraticelli, B.M.: A modification of benders' decomposition algorithm for discrete subproblems: An approach for stochastic programs with integer recourse. Journal of Global Optimization **22**(1-4), 319–342 (2002)
21. U.S. Energy Information Administration: Monthly energy review. Tech. rep., U.S. Energy Information Administration (2015)
22. Watson, J.P., Woodruff, D.L.: Progressive hedging innovations for a class of stochastic mixed-integer resource allocation problems. Computational Management Science **8**(4), 355–370 (2011)
23. Watson, J.P., Woodruff, D.L., Hart, W.E.: PySP: modeling and solving stochastic programs in Python. Mathematical Programming Computation **4**(2), 109–149 (2012)
24. Zhang, M., Küçükyavuz, S.: Finitely convergent decomposition algorithms for two-stage stochastic pure integer programs. SIAM Journal on Optimization **24**(4), 1933–1951 (2014)